TECHNIQUE FOR BIT-ACCURATE FILM GRAIN SIMULATION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority under 35 U.S.C. 119(e) to U.S. Provisional Patent Application Serial No. 60/511,026, filed on October 14, 2003, the teachings of which are incorporated herein.

TECHNICAL FIELD

This invention relates to a technique for simulating film grain in an image.

BACKGROUND ART

Motion picture films comprise silver-halide crystals dispersed in an emulsion, which is coated in thin layers on a film base. The exposure and development of these crystals form the photographic image consisting of discrete tiny particles of silver. In color negatives, tiny blobs of dye occur on the sites where the silver crystals form following chemical removal of the silver during development of the film stock. These small specks of dye commonly bear the label 'grain' in color film. Grain appears randomly distributed on the resulting image because of the random formation of silver crystals on the original emulsion. Within a uniformly exposed area, some crystals develop after exposure while others do not.

Grain varies in size and shape. The faster the film, the larger the clumps of silver formed and blobs of dye generated, and the more they tend to group together in random patterns. The term "granularity" typically refers to the grain pattern. The naked eye cannot distinguish individual grains, which vary from 0.0002 mm to about 0.002 mm. Instead, the eye resolves groups of grains, referred to as blobs. A viewer identifies these groups of blobs as film grain. As the image resolution becomes larger, the perception of the film grain becomes higher. Film grain becomes clearly noticeable on cinema and High Definition (HD) images, whereas film grain progressively loses importance in Standard Definition (SD) and becomes imperceptible in smaller formats.

Motion picture film typically contains image-dependent noise resulting either from the physical process of exposure and development of the photographic film or from the

subsequent editing of the images. Photographic film possesses a characteristic quasi-random pattern, or texture, resulting from physical granularity of the photographic emulsion. Alternatively, simulation of similar pattern can occur in computed-generated images in order to blend them with photographic film. In both cases, this image-dependent noise bears the designation of "film grain." Quite often, moderate grain texture presents a desirable feature in motion pictures. In some instances, the film grain provides visual cues that facilitate the correct perception of two-dimensional pictures. Film grade often varies within a single film to provide various clues as to time reference, point of view, etc. Many other technical and artistic demands exist for controlling grain texture in the motion picture industry. Therefore, preserving the grainy appearance of images throughout image processing and delivery chain has become a requirement in the motion picture industry.

Several commercially available products have the capability of simulating film grain, often for blending a computer-generated object into natural scene. Cineon® from Eastman Kodak Co, Rochester New York, one of the first digital film applications to implement grain simulation, produces very realistic results for many grain types. However, the Cineon® application does not yield good performance for many high speed films because of the noticeable diagonal stripes the application produces for high grain size settings. Further, the Cineon® application fails to simulate grain with adequate fidelity when images become subject to prior processing, for example, such as when the images are copied or digitally processed.

Another commercial product that simulates film grain is *Grain Surgery*™ from Visual Infinity Inc., which is used as a plug-in of Adobe ® After Effects ®. The *Grain Surgery*™ product appears to generate synthetic grain by filtering a set of random numbers. This approach suffers from disadvantage of a high computational complexity.

Thus, a need exists for an efficient film grain simulation technique, which reduces the need for memory bandwidth, and computational effort, thus permitting film grain simulation in cost-sensitive high volume devices, such as set top boxes.

BRIEF SUMMARY OF THE INVENTION

Briefly, in accordance with a preferred embodiment of the present principles, there is provided a method for simulating film grain in an image block of M x N pixels, where N and M are integers greater than zero. The method commences by first computing the average of the pixel values within the block of M x N pixels. A film grain block of M x N pixels is selected from among a pool of previously established blocks containing film grain as a function of the average value of the image block and a random number. Each pixel in the selected film grain block is blended with a corresponding pixel in the image block.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 depicts a block schematic drawing of an apparatus for generating pre-established film grain blocks for use in subsequent film grain simulation; and

FIGURE 2 depicts a block schematic drawing of an apparatus in accordance with the present principles for simulating film grain on a pixel-by-pixel basis using the pre-established film grain blocks generated by the apparatus of FIG. 1.

DETAILED DESCRIPTION

Introduction

The method of the present principles simulates film grain in accordance with film grain information transmitted with an image to which simulated grain is blended. In practice, the transmitted image typically undergoes compression (encoding) prior to transmission via one of a variety of well-known compression schemes, such as the H.264 compression scheme. With the transmitted image compressed using the H.264 compression scheme, transmission of the film grain information typically occurs via a Supplemental Enhancement Information (SEI) message. Pursuant to contributions recently adopted by the standards body responsible for promulgating the H.264 standard, the SEI message can now include various parameters that specify different film grain attributes.

Constraints on the film grain SEI message parameters

The method of the present principles imposes some constraints with regard to the number of parameters and their range of possible values allowed by the H.264 recommendation. TABLE 1 provides a list of such parameters, including a description of their semantics and the constraints imposed by the present principles.

TABLE I

FILM GRAIN PARAMETER	DESCRIPTION & CONSTRAINTS
model_id	This parameter specifies the simulation model. It shall be 0, which identifies the film grain simulation model as frequency filtering.
separate_colour_description_present_flag	This parameter specifies if the color space in which the parameters are estimated is different from the color space in which the video sequence (where the film grain SEI message has been embedded) has been encoded. It shall be 0, which identifies the

	color space for film grain the same than the encoded sequence.
blending_mode_id	This parameter identifies the blending mode used to blend the simulated film grain with the decoded images. It shall be 0, which correspond to an additive blending mode.
log2_scale_factor	This parameter identifies the logarithmic scale factor used to represent the film grain parameters in the SEI message. It shall be in the range [0, 4] to ensure film grain simulation can be performed using 16-bit arithmetic.
comp_model_present_flag[1]	This parameter enables the transmission of film grain parameters for the Cb color component in the YCbCr color space. It shall be 0, since film grain simulation in chroma is not supported.
comp_model_present_flag[2]	This parameter enables the transmission of film grain parameters for the Cr color component in the YCbCr color space. It shall be 0, since film grain simulation in chroma is not supported.
no_intensity_intervals_minus1[0]	This parameter defines the number of intensity intervals for which a specific set of parameters has been estimated. It shall be in the range [0, 7].
intensity_interval_lower_bound[0][i+1], intensity_interval_upper_bound[0][i]	These parameters define the boundaries of the luma intensity intervals for which different film grain parameters are defined. The lower bound of interval i+1 must be greater than the upper bound of interval i because multigenerational film grain is not allowed.

num_model_values_minus1[0]	This parameter specifies the number of model values present for each intensity interval in which the film grain has been modeled. It shall be in the range [0,4] because color correlation is not allowed.
comp_model_value[0][i][0]	This parameter represents the film grain intensity for each luminance intensity interval in which film grain has been modeled. It shall be in the range [0,255] to ensure film grain simulation can be performed using 16-bit arithmetic.

In addition to the previous constraints, the present principles imposes that film grain SEI messages precede I pictures, and only one film grain SEI message can precede a particular I picture. (The presence in the bit stream of **slice_type** equal to 7 or **nal_ref_idc** equal to 5, indicates an I picture.)

All the other parameters of the film grain SEI message have no constraint with respect to the standard specification.

Bit-accurate implementation of film grain simulation

Film grain simulation in accordance with the present principles occurs in a two-step process. First, generation of a pool of film grain blocks occurs during initialization, as described in greater detail with respect to FIG. 1. Thereafter, selected film grain portions are added to each luminance pixel of each decoded picture as described with respect to FIG. 2.

FIGURE 1 depicts an apparatus 10 in accordance with an illustrated embodiment of the present principles for generating a pool of film grain blocks for use in film grain simulation. The apparatus 10 typically generates a pool of 128 film grain blocks for each of as many as 8 different luminance intensity intervals. The SEI message field **num_intensity_intervals_minus1[0]** indicates one less than the number of the luminance intensity intervals.

The apparatus 10 accomplishes film grain noise initialization using a specified uniform pseudo-random number polynomial generator 12 and using a specified list of 2048 8-bit Gaussian distributed random numbers stored in a look-up table 14 The look-up table 14 stores random numbers in 2's complement form in the range [-63, 63]. The list of Gaussian random numbers appears in the Appendix.

According to the bit-accurate specification of the present principles, generation of the film grain blocks begins with the lowest luminance intensity interval. The uniform random number generator 12 generates an index for the Gaussian random number list stored in the look-up table 14 using a primitive polynomial modulo 2 operator, $x^{18} + x^5 + x^2 + x^1 + 1$. For ease of understanding, the term x(i, s) will indicate the i^{th} symbol of the sequence x, beginning with an initial seed s. The random number seed becomes reset to 1 upon the receipt of each film grain SEI message.

To form an individual 8x8 film grain block, a random block generator 16 reads 8 lines worth of 8 random numbers from the Gaussian random number look-up table 14. A random offset, from the random number generator 12, serves to access each line of 8 random numbers. Each line of the block produced by the block generator 16 is generated as following:

index = x(i, 1)
for n=0...7, B[i%8][n] = Gaussian_list[(index + n)%2048]
where i increments for each 8x1 block line.

The 8 x 8 block of random values read by the generator 16 undergoes a transform, typically an integer Discrete Cosine Transform (DCT), performed by an Integer DCT transform block 18. After the DCT transform, the 8 x 8 random values undergo frequency filtering at a frequency filter 20 in accordance with the cut frequencies specified in the SEI message. Following frequency filtering, the 8 x 8 random values undergo an inverse DCT transform by an inverse integer DCT block 22. A first scaling block 24 scales the pixels on the top and bottom block lines as follows:

for n=0..7, B'[0][n] = (B[0] [n] + 1) >> 1
for n=0..7, B'[7][n] = (B[7] [n] + 1) >> 1

This process continues until generation of a set of 128 film grain blocks for each luminance intensity interval. Following subsequent scaling by the second scaling block 26, the film grain blocks undergo storage in the film grain pool 28.

Block and Pixel Operations to Simulate Film Grain

FIGURE 2 illustrates an apparatus 200 in accordance with an illustrative embodiment of the present principles for simulating film grain on a pixel-by-pixel basis using the stored values in the film grain pool 28. The apparatus 200 includes a processing block 202 for creating an average of each 8 x 8 block of luma pixel values for comparison to the parameters **intensity_interval_lower_bound[0][i]** and **intensity_interval_upper_bound[0][i]** in the film grain SEI message to determine the correct luminance intensity interval for the current block.

A selector block 204 selects a k^{th} film grain block from the pool 28, using the random number generated by the uniform random number generator 16 from the polynomial modulo 128 as the block index. Thus, the noise generator 16, which generates uniformly distributed random numbers using a polynomial for the initialization process described with respect to FIG. 1, finds application in the apparatus 200 of FIG. 2 to select film grain blocks, with the random number seed reset to 1 after the pool creation process. If the resulting block index is identical to the previous one, the last bit of the index undergoes toggling. Such operation can occur using a bit-wise comparison and an XOR operator (^) as follows:

```
previous_index = index
index = x(k, 1) % 128
index ^= (index == previous_index)
```

Following block selection, a deblocking filter 206 deblocks the pixels on the right most column of the previously selected block and on the left most column of the current block. An adder 208 adds the deblocked film grain block to decoded luma pixels. (Since two horizontally adjacent blocks are required to perform deblocking, there is a 1-block delay between the block selected in 204 and the block added in 208.) A clipper 210 clips the result within the range [0, 255] for display. Note that film grain noise addition only occurs to luma pixels.

Scaling of Cut Frequencies

The parameters in the film grain SEI message of TABLE 1 assume the use of a 16 x 16 DCT in the simulation process. In particular, horizontal and vertical high cut frequencies, provided by **comp_model_value[0][i][1]** and **comp_model_value[0][i][2]**, and horizontal and vertical low cut frequencies, provided by **comp_model_value[0][i][3]** and **comp_model_value[0][i][4]**, serve to filter the transform coefficients of a block of 16x16 values.

In the illustrated embodiment, the use of 8 x 8 blocks will reduce complexity. Employing an 8 x 8 block transform using cut frequency parameters based on a 16 x 16 transform implies that all the cut frequencies require scaling before the grain generation.

The scaling of the cut frequencies occurs as follows:

$$comp\_model\_value'[0][i][j] = (comp\_model\_value\ [0][i][j] + 1) >> 1$$

where j is in the range [1,4]. Note that the scaling constitutes the equivalent of the integer division, rounded up to the nearest integer.

Integer Transform and Variance Scaling

The transform used for the frequency filtering corresponds to an 8 x 8 integer approximation to the DCT, using the following transformation matrix:

$$\mathbf{T}_8 = \begin{pmatrix} 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\ 8 & 7 & 4 & 2 & -2 & -4 & -7 & -8 \\ 7 & 3 & -3 & -7 & -7 & -3 & 3 & 7 \\ 7 & -2 & -8 & -4 & 4 & 8 & 2 & -7 \\ 6 & -6 & -6 & 6 & 6 & -6 & -6 & 6 \\ 4 & -8 & 2 & 7 & -7 & -2 & 8 & -4 \\ 3 & -7 & 7 & -3 & -3 & 7 & -7 & 3 \\ 2 & -4 & 7 & -8 & 8 & -7 & 4 & -2 \end{pmatrix}$$

16-bit arithmetic can be used. The forward integer transformation of a given block of random noise is defined as:

$$\tilde{\mathbf{B}} = (((\mathbf{T}_8 \times \mathbf{B} + 8) >> 4) \times \mathbf{T}_8^{\,T} + 8) >> 4$$

with 11 bits used for $\tilde{\mathbf{B}}$.

The inverse integer transform is defined as:

$$\mathbf{B} = (\mathbf{T}_8^{\,T} \times \tilde{\mathbf{B}} \times \mathbf{T}_8 + 128) >> 8$$

with 8 bits used for B.

Following the inverse transform, the block B undergoes scaling as follows, assuming it is in the k^{th} luminance intensity interval,

$$val = B(i, j) * comp\_model\_value[0][k][0]$$

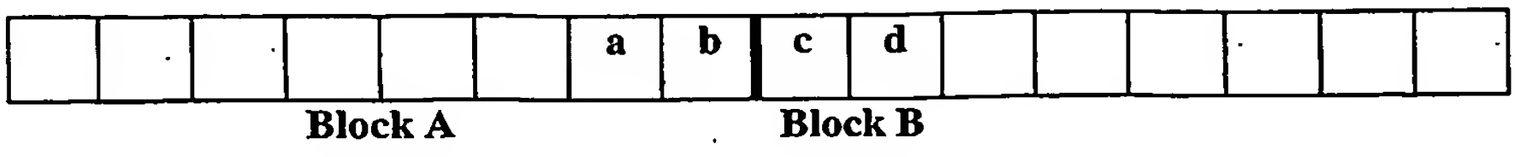
$$B'(i, j) = (((val - (val >>4) + 2^{log2\_scale\_factor - 1}) >> log2\_scale\_factor) + 16) >> 5$$

where the operation (val - (val>>4)) compensates the scaling of the integer transform; log2_scale_factor, transmitted in the SEI message, scales comp_model_value[0][k][0]; and 5 scales the Gaussian numbers provided in the Appendix.

Deblocking filter 206

As indicated, the film grain simulation apparatus of FIG. 2 includes a deblocking filter 206 for smoothing blocking artifacts resulting from the small size of the transform. In

the illustrated embodiment, the deblocking filter 206 takes the form of a 3-tap filter applied to all pixels bordering the 8x8 block left and right edges. Given a row of pixels belonging to two adjacent 8x8 blocks, the transition between blocks being located between pixels b and c,



application of the deblocking occurs as follows:

$$b' = (a + (b << 1) + c) >> 2$$

$$c' = (b + (c << 1) + d) >> 2$$

where b' and c' replace the value of the original pixels b and c, respectively. Deblocking of the left and right block edges occurs for every film grain block before addition to the decoded image.

The foregoing describes a technique for simulating film grain in an image.

1

Appendix

The list of the 2048 Gaussian distributed random numbers are:

char Gaussian[2048] = {

0xFB, 0x05, 0x33, 0xFB, 0x14, 0xEF, 0x06, 0x1D, 0x26, 0x30, 0xD5, 0x01, 0x20, 0xD9, 0x16, 0x1B,
0xE7, 0x0A, 0x06, 0xFB, 0xF6, 0xF7, 0x10, 0xC1, 0x08, 0xFE, 0xCC, 0x09, 0x09, 0x23, 0x17, 0xFB,
0xED, 0x15, 0xFF, 0x25, 0xDF, 0x1A, 0xD3, 0x10, 0xE9, 0x0A, 0xFF, 0xE5, 0x18, 0x00, 0xE4, 0xEC,
0x00, 0x3C, 0xC1, 0xCB, 0xE8, 0x04, 0x07, 0x3F, 0x3D, 0x36, 0x19, 0x3F, 0x00, 0x03, 0x38, 0x09,
0x0E, 0x06, 0x26, 0x38, 0x28, 0xE2, 0xC1, 0x37, 0xE7, 0xF2, 0x01, 0xE8, 0xF5, 0x1D, 0xF2, 0xDC,
0x05, 0x38, 0x21, 0x27, 0xFF, 0xC7, 0xD5, 0xFE, 0xFE, 0x14, 0x1D, 0xD8, 0x18, 0xF3, 0xF1, 0xEF,
0xCC, 0x19, 0x08, 0xF4, 0xEF, 0xFA, 0xF9, 0xC1, 0xE5, 0xF5, 0xE5, 0xC1, 0xC8, 0x02, 0xF4, 0xDC,
0x3F, 0x3F, 0xFF, 0x14, 0x2B, 0xE0, 0xF9, 0x1B, 0x09, 0x2D, 0xD8, 0xE0, 0xE0, 0x11, 0xFD, 0xE5,
0x31, 0xFD, 0x2C, 0x3E, 0xF3, 0x2D, 0x00, 0x1F, 0x1D, 0xF9, 0xF5, 0x38, 0xF0, 0x3A, 0x06, 0x0C,
0x19, 0xF8, 0x35, 0xFD, 0x1A, 0x13, 0xEF, 0x08, 0xFD, 0x02, 0xD3, 0x03, 0x1F, 0x1F, 0xF9, 0x13,
0xEE, 0x09, 0x1B, 0x08, 0xE7, 0x13, 0x10, 0xEE, 0x3E, 0xED, 0xC5, 0x08, 0xF1, 0x00, 0x09, 0x31,
0x1E, 0x32, 0xFA, 0xDC, 0xF8, 0xE7, 0x31, 0x01, 0x01, 0x1D, 0x10, 0xFF, 0xFF, 0x04, 0xEC, 0xCC,
0xEE, 0x06, 0x3F, 0x07, 0xC1, 0xF1, 0xD5, 0xED, 0xE5, 0x16, 0xEC, 0x25, 0x0B, 0xF7, 0xF5, 0xDD,
0x25, 0xE6, 0x00, 0x10, 0xEA, 0x08, 0xD2, 0x1D, 0xE0, 0xDF, 0x1B, 0xCE, 0xF2, 0xD5, 0xEF, 0xD2,
0x21, 0x02, 0xDC, 0xE2, 0x2E, 0xEB, 0x06, 0xF4, 0xEE, 0xC1, 0xF8, 0x07, 0xC1, 0x1F, 0x11, 0x0F,
0x2E, 0x08, 0xE7, 0xE3, 0x23, 0x26, 0x28, 0x3F, 0x3F, 0x1E, 0x10, 0xCC, 0xD2, 0x00, 0x00, 0x25,
0xDE, 0x23, 0x3F, 0xF7, 0xC9, 0x0E, 0x0B, 0x07, 0x01, 0x13, 0x2D, 0x02, 0x14, 0x00, 0xFE, 0x13,
0x07, 0x38, 0xF2, 0xEE, 0x19, 0x15, 0x35, 0x0D, 0x3B, 0x03, 0xD9, 0x0C, 0xDE, 0xF6, 0x2E, 0xFB,
0x00, 0x09, 0x14, 0xE7, 0x27, 0xC1, 0xEB, 0x3F, 0x08, 0x05, 0xF6, 0x0F, 0xE7, 0x0D, 0xD4, 0xD3,
0xED, 0xF7, 0xFC, 0x0C, 0xC6, 0x23, 0xF4, 0xEB, 0x00, 0x05, 0x2A, 0xCB, 0x13, 0xF0, 0xC1, 0x17,
0x19, 0xF4, 0xF6, 0x16, 0x00, 0x07, 0xEF, 0xDE, 0x00, 0xDC, 0x0C, 0xFD, 0x00, 0x0E, 0xFF, 0x16,
0x10, 0xF0, 0x3A, 0xEA, 0x27, 0xF5, 0xF8, 0xCA, 0xFB, 0xDD, 0x2C, 0xE9, 0x0B, 0xD3, 0x3B, 0xEE,
0x18, 0xC1, 0x1D, 0x10, 0xD8, 0xFB, 0xF8, 0xFD, 0x16, 0xC1, 0xF9, 0x2C, 0x3F, 0x08, 0x31, 0xED,
0xF0, 0x12, 0x15, 0xED, 0xF1, 0xF6, 0x34, 0xF7, 0x09, 0x09, 0xE3, 0xFC, 0x0F, 0x00, 0xC1, 0x10,
0x3F, 0xD6, 0x25, 0x0B, 0xEC, 0xE8, 0xC1, 0xCB, 0xF9, 0x16, 0xDB, 0x00, 0x0E, 0xF7, 0x14, 0xDE,
0xED, 0x06, 0x3F, 0xFF, 0x02, 0x0A, 0xDC, 0xE3, 0xC1, 0xFF, 0xFF, 0xE6, 0xFE, 0xC5, 0x2E, 0x3B,
0xD8, 0xE8, 0x00, 0x09, 0xEA, 0x21, 0x26, 0xFA, 0xF6, 0xC1, 0x11, 0xEC, 0x1B, 0x3B, 0xFE, 0xC7,
0xF5, 0x22, 0xF9, 0xD3, 0x0C, 0xD7, 0xEB, 0xC1, 0x35, 0xF4, 0xEE, 0x13, 0xFD, 0xFD, 0xD7, 0x02,
0xD5, 0x15, 0xEF, 0x04, 0xC1, 0x13, 0x22, 0x18, 0xE1, 0x24, 0xE8, 0x36, 0xF3, 0xD4, 0xE9, 0xED,
0x16, 0x18, 0xFF, 0x1D, 0xEC, 0x28, 0x04, 0xC1, 0xFC, 0xE4, 0xE8, 0x3E, 0xE0, 0x17, 0x11, 0x3A,
0x07, 0xFB, 0xD0, 0x36, 0x2F, 0xF8, 0xE5, 0x22, 0x03, 0xFA, 0xFE, 0x18, 0x12, 0xEA, 0x3C, 0xF1,
0xDA, 0x14, 0xEA, 0x02, 0x01, 0x22, 0x08, 0xD9, 0x00, 0xD9, 0x02, 0x3F, 0x15, 0x0D, 0x3F, 0xC1,
0x0D, 0xE5, 0xF3, 0x1B, 0x37, 0x17, 0x35, 0x00, 0xDA, 0x00, 0x1A, 0xFC, 0xF5, 0xEB, 0x3D, 0x36,
0x3F, 0x32, 0x21, 0x17, 0x02, 0x00, 0x3D, 0xFA, 0xE5, 0xF0, 0xE8, 0x2C, 0x20, 0xCC, 0xFE, 0x2F,

0xE6, 0x1F, 0x16, 0x0E, 0x17, 0x09, 0xEF, 0x07, 0x14, 0x17, 0xD0, 0xF4, 0x2F, 0xDB, 0x3F, 0xC7, 0x3F, 0xDF, 0x00, 0xF8, 0x19, 0xD1, 0x17, 0x05, 0x11, 0xEA, 0xDB, 0x2C, 0xCB, 0xFC, 0xE4, 0xF2, 0xCA, 0xF4, 0x3F, 0xE2, 0xFA, 0x26, 0xEA, 0x08, 0x09, 0x29, 0xF5, 0x04, 0x3F, 0xDF, 0x1A, 0x01, 0x0C, 0x06, 0x37, 0x15, 0xC8, 0xF5, 0x05, 0xF4, 0x29, 0x21, 0xFA, 0x25, 0xC3, 0x1D, 0x3F, 0xFB, 0x31, 0xF7, 0x1F, 0xED, 0x1A, 0x04, 0x03, 0x1E, 0xE5, 0x01, 0xE4, 0x38, 0xCC, 0xE3, 0x01, 0xFC, 0xE9, 0x24, 0x2A, 0xE5, 0xEF, 0x06, 0x3B, 0x0D, 0x2E, 0xDD, 0x06, 0xCF, 0xDD, 0xF6, 0x0E, 0x23, 0xD1, 0x09, 0xE6, 0x20, 0xFA, 0xE1, 0xF4, 0x20, 0x24, 0xFC, 0x3F, 0x00, 0xC1, 0x33, 0xF6, 0xDC, 0xC9, 0xCD, 0xFD, 0x0E, 0xEC, 0xF6, 0xE3, 0xF2, 0xF4, 0x09, 0xFE, 0xE7, 0x2F, 0xE3, 0xD1, 0xEE, 0x11, 0x09, 0xDE, 0x3F, 0xF7, 0xC1, 0xF5, 0xC5, 0xE6, 0x12, 0x25, 0xC1, 0x00, 0xFB, 0xC5, 0xE6, 0xF3, 0x13, 0x22, 0x08, 0x08, 0xC7, 0x2C, 0x1F, 0x0C, 0x12, 0xF5, 0x18, 0xCE, 0xF1, 0xFC, 0xD1, 0xE6, 0x02, 0x2E, 0xF5, 0xE8, 0xFC, 0x19, 0x01, 0xDB, 0xD4, 0xFB, 0xED, 0x3F, 0xD5, 0xF5, 0x09, 0x0A, 0x38, 0x25, 0x19, 0xF1, 0x2E, 0xE1, 0x03, 0xFB, 0x17, 0x12, 0x32, 0xEB, 0xF8, 0xE6, 0xFD, 0xEE, 0xDA, 0xF1, 0xF6, 0x1F, 0x0F, 0x1F, 0x0A, 0xC1, 0x0F, 0x1F, 0x12, 0x33, 0xD6, 0xFC, 0x26, 0x27, 0x1D, 0xD9, 0xFD, 0x11, 0x04, 0x28, 0xF4, 0xFC, 0x01, 0xF8, 0x23, 0x3F, 0x29, 0xD5, 0x1B, 0x09, 0xC5, 0xC3, 0x12, 0x05, 0x3F, 0x1C, 0xE5, 0x38, 0x06, 0x0C, 0x10, 0xFA, 0xE9, 0x0A, 0xFA, 0x02, 0x1C, 0x0D, 0x0C, 0x0C, 0xFB, 0xEE, 0x12, 0xD2, 0x26, 0x28, 0x04, 0x19, 0x06, 0x21, 0xFA, 0x00, 0x10, 0x16, 0xDB, 0x10, 0xED, 0xF5, 0xE8, 0xC1, 0xF3, 0x0F, 0xFC, 0x11, 0x06, 0x23, 0x06, 0x1C, 0x05, 0xE6, 0xD6, 0x1A, 0xEA, 0xEF, 0x00, 0x3F, 0x05, 0xDF, 0xEA, 0x17, 0xC7, 0x01, 0x05, 0x1C, 0xEF, 0x3B, 0xF7, 0xE2, 0x1A, 0xE3, 0xC1, 0xE8, 0xF5, 0x01, 0xFE, 0x08, 0xD8, 0xFE, 0x3F, 0x0C, 0x27, 0x21, 0x1F, 0xF4, 0x06, 0xE0, 0xEE, 0xC1, 0xF2, 0x0A, 0xE1, 0x20, 0xE6, 0xEC, 0x36, 0xE1, 0x07, 0xF6, 0x06, 0x0E, 0xE1, 0x0A, 0x0D, 0x2F, 0xEA, 0xE3, 0xC6, 0xFC, 0x27, 0xE8, 0x0B, 0xEB, 0xF8, 0x17, 0xE9, 0xC4, 0xEF, 0xF2, 0xE6, 0xEA, 0x0E, 0x3F, 0xFA, 0x18, 0xFC, 0xC1, 0x25, 0xF3, 0xF5, 0x2C, 0x1D, 0x05, 0xD1, 0x28, 0xE3, 0x1D, 0x1E, 0xF4, 0x14, 0xD3, 0xFF, 0xF6, 0xE3, 0xEA, 0xE3, 0xF5, 0xE6, 0x23, 0xF2, 0x21, 0xF1, 0xF5, 0x07, 0xF8, 0xDF, 0xF4, 0xF2, 0xE2, 0x17, 0x12, 0x08, 0x07, 0xEE, 0xF5, 0xFB, 0x04, 0xF3, 0xF7, 0x1D, 0x16, 0xE8, 0xE9, 0xFF, 0xF6, 0xD8, 0x0E, 0xDF, 0xC1, 0x25, 0x32, 0x02, 0xF8, 0x30, 0x11, 0xE0, 0x14, 0xE7, 0x03, 0xE3, 0x0B, 0xE4, 0xF7, 0xF4, 0xC5, 0xDC, 0x2D, 0x07, 0xF9, 0x27, 0xF0, 0xD9, 0xC1, 0xEF, 0x14, 0x26, 0xD7, 0x00, 0x1B, 0x0B, 0xDB, 0x3F, 0xF8, 0xF6, 0x06, 0x0F, 0x1B, 0xC8, 0xC1, 0x2C, 0x1B, 0x1E, 0x06, 0x1B, 0xFA, 0xC8, 0xF9, 0x0F, 0x18, 0xDF, 0xF8, 0x2D, 0xFC, 0x00, 0x0A, 0x22, 0xDD, 0x31, 0xF7, 0xC8, 0x20, 0xD3, 0xFC, 0xFC, 0xDD, 0x3F, 0x19, 0xD8, 0xE8, 0x0C, 0x1E, 0xE2, 0xC9, 0x03, 0xEC, 0x3F, 0x2B, 0xE0, 0x35, 0xC1, 0xFE, 0x11, 0xF9, 0x14, 0xE8, 0x06, 0x06, 0x24, 0xCE, 0xF3, 0x26, 0x3F, 0xFD, 0xCE, 0x2C, 0x12, 0x3C, 0x2C, 0xC2, 0xE3, 0x06, 0xD2, 0xC7, 0x0A, 0xDF, 0xD5, 0xD1, 0xC5, 0x15, 0xF2, 0xF1, 0x08, 0x02, 0xE6, 0xE2, 0x0A, 0xEB, 0x05, 0xDA, 0xE3, 0x06, 0x0E, 0x01, 0x03, 0xDC, 0x13, 0xE3, 0xFB, 0x36, 0xE6, 0x14, 0x21, 0xFA, 0xC1, 0xC1, 0xE8, 0x0B, 0x0E, 0x17, 0x11, 0x2D, 0x11, 0xF0, 0x39, 0xE7, 0xF0, 0xE7, 0x2D, 0x03, 0xD7, 0x24, 0xF4, 0xCD, 0x0C, 0xFB, 0x26, 0x2A, 0x02, 0x21, 0xD8, 0xFA, 0xF8, 0xF0, 0xE8, 0x09, 0x19, 0x0C, 0x04, 0x1F, 0xCD, 0xFA, 0x12, 0x3F, 0x38, 0x30, 0x11, 0x00, 0xF0, 0xE5, 0x3F, 0xC3, 0xF0, 0x1E, 0xFD, 0x3B, 0xF0, 0xC1, 0xE6, 0xEB, 0x1F, 0x01, 0xFE, 0xF4, 0x23, 0xE4, 0xF0, 0xEB, 0xEB, 0x10, 0xE4, 0xC1, 0x3F, 0x0C, 0xEF, 0xFB, 0x08, 0xD8, 0x0E, 0xE4, 0x14, 0xC1, 0xC1, 0x0A, 0xE9, 0xFB, 0xEF, 0xE1, 0xE7, 0xF0, 0xD8, 0x27, 0xDA, 0xDC, 0x04, 0x0D, 0xDC, 0xFC, 0xDB, 0xD6, 0xD6, 0xE4, 0x0C, 0x27, 0xFC, 0xD0, 0x11,

0xE0, 0x04, 0xE3, 0x07, 0x00, 0xEC, 0x10, 0xD5, 0xEA, 0x08, 0xFF, 0xFC, 0x1D, 0x13, 0x05, 0xCA, 0xED, 0x0B, 0x10, 0x08, 0xF2, 0x01, 0x19, 0xCA, 0xFE, 0x32, 0x00, 0x20, 0x0B, 0x00, 0x3F, 0x1E, 0x16, 0x0C, 0xF1, 0x03, 0x04, 0xFD, 0xE8, 0x31, 0x08, 0x15, 0x00, 0xEC, 0x10, 0xED, 0xE6, 0x05, 0xCA, 0xF7, 0x1C, 0xC1, 0x22, 0x0D, 0x19, 0x2E, 0x13, 0x1E, 0xE7, 0x16, 0xED, 0x06, 0x2A, 0x3C, 0x0D, 0x21, 0x16, 0xC9, 0xD7, 0xFF, 0x0F, 0x12, 0x09, 0xEE, 0x1D, 0x23, 0x13, 0xDA, 0xE9, 0x1D, 0xD9, 0x03, 0xE1, 0xEF, 0xFA, 0x1E, 0x14, 0xC1, 0x23, 0xFE, 0x0B, 0xE5, 0x19, 0xC1, 0x21, 0xFE, 0xEC, 0x0E, 0xE1, 0x1D, 0xFF, 0x00, 0xF7, 0xEA, 0xD2, 0xD8, 0xD0, 0xF9, 0xE6, 0xFB, 0xFB, 0xDA, 0x06, 0x00, 0x03, 0xDF, 0xC1, 0x3F, 0xF3, 0x0D, 0xFA, 0x08, 0xFA, 0xF3, 0x00, 0x04, 0xE9, 0xF0, 0xF9, 0x0D, 0xF1, 0xE3, 0x1D, 0x26, 0xC4, 0x0D, 0x13, 0xE5, 0xE1, 0xF1, 0xF6, 0xEE, 0xF1, 0xFD, 0xC1, 0xF4, 0xE2, 0x23, 0xC1, 0x38, 0xC1, 0x3F, 0x2B, 0xFD, 0x39, 0x36, 0x1A, 0x2B, 0xC1, 0x01, 0x07, 0x0B, 0x25, 0xCC, 0xE7, 0x01, 0x24, 0xD8, 0xC9, 0xDB, 0x20, 0x28, 0x0C, 0x1A, 0x3F, 0xEA, 0xE7, 0xCD, 0xEC, 0xE0, 0xF2, 0x27, 0xDF, 0x20, 0xF0, 0xF1, 0xFD, 0x3F, 0x00, 0xFA, 0xE7, 0x21, 0xF9, 0x02, 0xD2, 0x0E, 0xEF, 0xFD, 0xD3, 0xE4, 0xFF, 0x12, 0x15, 0x16, 0xF1, 0xDE, 0xFD, 0x12, 0x13, 0xE7, 0x15, 0xD8, 0x1D, 0x02, 0x3F, 0x06, 0x1C, 0x21, 0x16, 0x1D, 0xEB, 0xEB, 0x14, 0xF9, 0xC5, 0x0C, 0x01, 0xFB, 0x09, 0xFA, 0x19, 0x0E, 0x01, 0x1B, 0xE8, 0xFB, 0x00, 0x01, 0x30, 0xF7, 0x0E, 0x14, 0x06, 0x15, 0x27, 0xEA, 0x1B, 0xCB, 0xEB, 0xF7, 0x3F, 0x07, 0xFB, 0xF7, 0xD8, 0x29, 0xEE, 0x26, 0xCA, 0x07, 0x20, 0xE8, 0x15, 0x05, 0x06, 0x0D, 0x0D, 0x1E, 0x1C, 0x0F, 0x0D, 0x35, 0xF7, 0x1B, 0x06, 0x30, 0x02, 0xFD, 0xE2, 0xCD, 0x2F, 0x35, 0xEB, 0x1A, 0x0D, 0xE9, 0xFC, 0x34, 0xE6, 0x17, 0x2C, 0x33, 0xF0, 0x13, 0xEF, 0x1B, 0x19, 0x23, 0xD1, 0xEF, 0xD5, 0xCB, 0xF7, 0xF1, 0x04, 0xF7, 0x27, 0xF9, 0x26, 0x02, 0xF7, 0xCB, 0x2A, 0x0A, 0xEA, 0xED, 0xEC, 0x04, 0xF2, 0x25, 0x17, 0xDB, 0x1E, 0xC1, 0x3C, 0xC9, 0xE4, 0xF1, 0x14, 0x03, 0x27, 0x25, 0x21, 0x1C, 0x14, 0xF4, 0x0F, 0x12, 0xE9, 0xEE, 0x15, 0xDC, 0xEE, 0x1F, 0x3F, 0xDE, 0xE7, 0x2C, 0xF0, 0xE2, 0x1D, 0xE5, 0x15, 0x07, 0x02, 0xDF, 0x06, 0xD3, 0x1F, 0x0E, 0xED, 0xFF, 0x29, 0xFF, 0xED, 0xD6, 0xD6, 0x1C, 0x11, 0xDE, 0xE2, 0x0E, 0xEE, 0xD1, 0xD9, 0x02, 0x0F, 0xFE, 0xF0, 0xD9, 0xF6, 0xFC, 0xDA, 0x16, 0x03, 0xD2, 0xDD, 0x20, 0x04, 0xE8, 0x3F, 0xDE, 0x0C, 0xFB, 0xED, 0xC7, 0x1F, 0xC1, 0xCE, 0x02, 0xF1, 0x37, 0x0B, 0xE3, 0x20, 0xCE, 0x0D, 0xEB, 0x0A, 0xE3, 0xF3, 0xDC, 0x01, 0xD2, 0x02, 0x3F, 0x02, 0x25, 0xD5, 0xFC, 0xEB, 0xCE, 0x3F, 0x00, 0x3E, 0x2D, 0xE1, 0x19, 0x1C, 0x01, 0x28, 0xC1, 0x3F, 0x27, 0x3F, 0xF2, 0x0E, 0x3A, 0xDB, 0xF8, 0xE4, 0x34, 0x18, 0x16, 0x0C, 0xDD, 0x18, 0xED, 0xCB, 0x0F, 0xF0, 0x01, 0xFB, 0x14, 0xC1, 0x19, 0xCC, 0xEB, 0xEE, 0x19, 0x00, 0x17, 0x2B, 0xFC, 0x26, 0x0D, 0xEC, 0xF4, 0x2D, 0x2B, 0xE5, 0x25, 0x05, 0x10, 0x26, 0x1D, 0x3F, 0x3F, 0xFD, 0xDC, 0x18, 0xF0, 0xCB, 0xEF, 0x12, 0x1C, 0x1A, 0xF8, 0xFE, 0x29, 0x1A, 0xCB, 0x1A, 0xC2, 0x0E, 0x0B, 0x1B, 0xEB, 0xD5, 0xF8, 0xFD, 0x17, 0x0B, 0xFC, 0x00, 0xFA, 0x37, 0x25, 0x0D, 0xE6, 0xEE, 0xF0, 0x13, 0x0F, 0x21, 0x13, 0x13, 0xE1, 0x12, 0x01, 0x0A, 0xF1, 0xE7, 0xF3, 0x1A, 0xED, 0xD5, 0x0A, 0x19, 0x39, 0x09, 0xD8, 0xDE, 0x00, 0xF9, 0xE9, 0xEA, 0xFF, 0x3E, 0x08, 0xFA, 0x0B, 0xD7, 0xD7, 0xDE, 0xF7, 0xE0, 0xC1, 0x04, 0x28, 0xE8, 0x1E, 0x03, 0xEE, 0xEA, 0xEB, 0x1C, 0xF3, 0x17, 0x09, 0xD6, 0x17, 0xFA, 0x14, 0xEE, 0xDB, 0xE2, 0x2A, 0xD9, 0xC1, 0x05, 0x19, 0x00, 0xFF, 0x06, 0x17, 0x02, 0x09, 0xD9, 0xE5, 0xF3, 0x20, 0xDD, 0x05, 0xCB, 0x09, 0xF8, 0x05, 0xF1, 0x1F, 0xE5, 0x12, 0x25, 0xF8, 0x3F, 0xDC, 0xF0, 0xF2, 0xC5, 0x34, 0x21, 0x35, 0xCD, 0xCC, 0x23, 0x1E, 0x01, 0x0B, 0xFF, 0x10, 0xFE, 0xF9, 0xDF, 0xF9, 0xF5, 0xE5, 0x07, 0xE1, 0x25, 0x1C, 0xC9, 0x00, 0x29, 0xF3, 0x0A, 0x25, 0xED, 0xF8, 0xFB, 0x20, 0xF8, 0xC1, 0xE5, 0xE0, 0x0F, 0x2F, 0x3A, 0x01, 0xC8, 0xFD,

0xCA, 0xE1, 0x30, 0x04, 0x19, 0x03, 0x25, 0xF3, 0x24, 0x38, 0xEE, 0xC9, 0x2F, 0xE7, 0x0B, 0xFA,
0xF7, 0x1B, 0x0A, 0x0B, 0x2D, 0x2D, 0x0B, 0xE8, 0x08, 0xDB, 0x0B, 0x04, 0xE8, 0xD0, 0xEE, 0x18,
0xEF, 0x11, 0xC1, 0xD6, 0x15, 0x3F, 0xF5, 0xF4, 0x2A, 0x29, 0xEF, 0xF0, 0xFA, 0x36, 0x33, 0xED,
0x19, 0xDF, 0x11, 0x09, 0xF5, 0x18, 0xF1, 0x3F, 0x14, 0x0C, 0xD2, 0xFF, 0xFF, 0x34, 0x01, 0xE4,
0xF8, 0x03, 0x3F, 0xF8, 0x3E, 0x21, 0x22, 0xE2, 0x0F, 0xEF, 0x1A, 0xE4, 0xF5, 0x08, 0x15, 0xEF,
0xF3, 0xE4, 0xDF, 0xF6, 0xFC, 0xE8, 0x21, 0x06, 0x20, 0x02, 0x17, 0x1B, 0x3F, 0xDB, 0x16, 0x2C,
0xE0, 0xFA, 0xDA, 0xD8, 0xD3, 0x0B, 0x0E, 0x10, 0xED, 0xD5, 0xF0, 0x30, 0xD3, 0x13, 0x04, 0xE1,
0xFF, 0xFB, 0x3F, 0xE8, 0xEE, 0xE5, 0x0B, 0xEF, 0xEF, 0xE6, 0x2C, 0xD3, 0x00, 0x18, 0x26, 0xFE,
0xC1, 0x08, 0x16, 0xFE, 0xDC, 0x00, 0xE4, 0xF7, 0xDC, 0x0E, 0x2E, 0x1D, 0x18, 0x0A, 0x08, 0x37,
0xC9, 0x10, 0xD7, 0x17, 0x17, 0xFB, 0x11, 0xD5, 0x15, 0x1C, 0xD0, 0x3F, 0xF8, 0x00, 0x00, 0xED,
0xC1, 0xFF, 0x00, 0x1F, 0x2E, 0x00, 0x12, 0xE0, 0xE2, 0xF7, 0x13, 0xC1, 0x1C, 0x18, 0xF8, 0x3F,
0x2C, 0xEB, 0xCA, 0xE7, 0xF8, 0x03, 0xEE, 0x22, 0x17, 0xF9, 0x35, 0x14, 0x1C, 0x03, 0x09, 0x03,
0x01, 0x2B, 0xD4, 0xD2, 0xF8, 0xF6, 0xF5, 0x06, 0x03, 0xFE, 0xDA, 0xD3, 0xFF, 0x03, 0xEF, 0xFE,
0x09, 0x01, 0xC9, 0x02, 0xDF, 0xD8, 0x3C, 0xF7, 0xF0, 0xEE, 0xD6, 0x3F, 0x21, 0x16, 0x08, 0x17
};